CS695

# introduction to GPUs & GPU virtualization

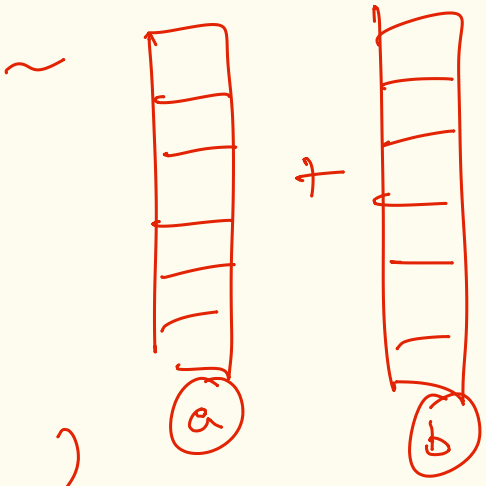
~ GPUs

(i) SIMD

Single instruction multiple data.

tasks

logic

~ 


1 thread

```
loop {
    a[i], b[i]
}
```

Quiz 2

Fri. 11/4

8:30 am

Faas platform hands-on

A4

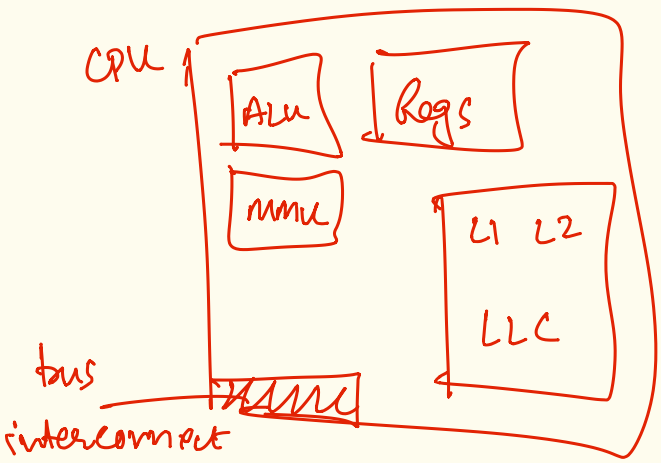
~ assign a paper reading material

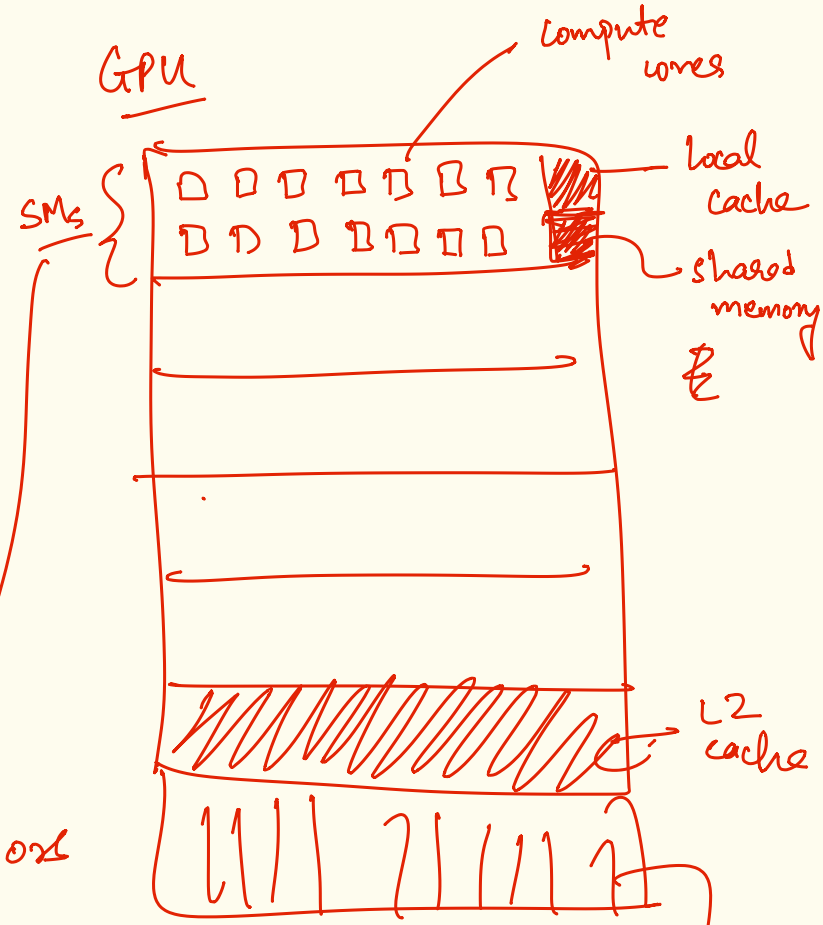
~ submit 5-6 slides of content on the paper.

a[i]+b[i] ~ same operation on all threads of execution.

thread specific

(ii) GPU architecture




#NVIDIA

↳ CUDA API

↳ programming framework

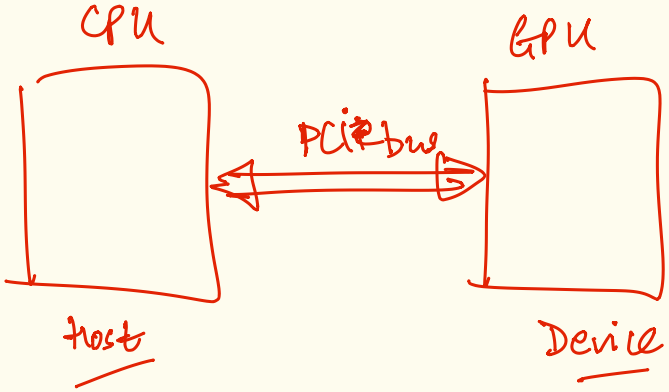
- threads ~ instance of parallel execution
  - ↳ scheduled on a core
- blocks ~ sets of threads
  - ↳ schedulable granularity on an SM.
- grids
  - ↳ #blocks of your program.

CUDA programs

H/w → discrete GPAs (i)

integrated / HSA GPUs. (ii)

(i)




(ii) GPU & CPU share a memory bus

⇒ both can same access phy. addresses

copy mem from H to D

do GPU work ——— NVIDIA cuda Kernel

copy mem from D to H

(#) anatomy CUDA program

```
__global__ add (int* in, int* out)
{
    tid = (blockid * blocksize + threadid)
    a[tid] = a[tid] + b[tid];
}
```

<<< 2, 2 >>>
2 2

block 0 tid 0
1

block 1 tid 0
1

✓ <<< 2, 2 >>>

(*) <<< 1, 4 >>>

```
main ( ) {
    int a[ ], b[ ];
    ga = cudamalloc ( );
    gb = cudamalloc ( );
    copy mem H to D (a, ga)
```

→ <<< #blocks, #threads per block >>> add (ga, gb); (b, gb);

```
}
```

(*) appln's.

- vector-based arithmetic
  - AI/ML
  - image processing
  - audio/video analysis
  - cryptography

} virtualization

- sharing
- multiplexing
- utilization
- performance

are at a higher imp. level for GPUs.

(#) GPU virtualization

- MiG } Multi-Instance GPU
- vGPU } vGPU
  - h/w assisted & GPU driver
  - breakup 1 GPU into * GPUs (device interface)
    - MiG – spatial multiplexing
    - vGPU – temporal multiplexing
      - memory is divided spatially
      - cores divided/multiplexed temporally
        - best-effort ~ if vGPU has no work schedule from other vGPUs.
- MPS ~ slice ~ gpu driver provided per process GPU resource
- (x) API remoting / virtualization
- (x) Kernel slicing
- FaaS

4/7 & 3/7 — all GPU resources divided spatially in this ratio